Refine Search

Search Results -

Terms	Documents
L1 AND DLL	24

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L2

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Monday, December 27, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=NO; OP=OR			
L2	L1 AND DLL	24	L2
L1	717/165,171,176.ccls.	247	L1

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 24 of 24 returned.

☐ 1. Document ID: US 6832371 B1

L2: Entry 1 of 24 File: USPT Dec 14, 2004

US-PAT-NO: 6832371
DOCUMENT-IDENTIFIER: US 6832371 B1

TITLE: Method for automatically updating a computer registry

DATE-ISSUED: December 14, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hussey; Thomas E.	Bothell	WA		

US-CL-CURRENT: 717/165; 713/100, 719/321, 719/331

ABSTRACT:

In a computer system, a method for automatically registering resources required for an application program module to execute. After the application program module is booted, a registration cache is examined to determine its status. The registration cache is stored in association with the application program module and it indicates whether a registry on the computer system needs to be updated, such as after the user has moved files or renamed files such that registry keys in the registry may no longer be valid. If the registration cache indicates that the registry needs to be updated, then a search is made through a predetermined directory, such as the application program module folder. The search is looking for an application file or a dynamic link library file. Upon detecting an application file or a dynamic link library file, then a resource fork in association with the file is opened and it is determined whether there is a self-registration resource in the resource fork. If so, then the self-registration resource is registered by initiating a self-registration dynamic link library. It is also determined whether there is an Object Linking and Embedding (OLE) Type Library (OTLB) resource in the resource fork and if so, then registering the OTLB resource by initiating an OLE call.

13 Claims, 4 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 2. Document ID: US 6789255 B1

L2: Entry 2 of 24 File: USPT Sep 7, 2004

US-PAT-NO: 6789255
DOCUMENT-IDENTIFIER: US 6789255 B1

TITLE: Determining update availability via set intersection over a sub-optimal pathway

DATE-ISSUED: September 7, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Pedrizetti; Raymond D.	Issaquah	WA		
Quinn; Scott D.	Issaquah	WA		
Bragg; Timothy W.	Redmond	WA		

US-CL-CURRENT: 717/169; 717/170, 717/171

ABSTRACT:

A low bandwidth link can be used optimally for software updates, by successively transferring more information about the updates only as the likelihood of an applicable update successively increases. A many-to-one mapping function (e.g. a hash function) is applied to update identifiers on a server to generate a table of single bit entries corresponding to the updates. At a client, the same mapping function is applied to program identifiers to determine whether the server has a potential update. If a potential update is noted, a second transmission is requested for conveying additional data from the server by which hash collisions can be identified. A third transmission from the server is received conveying the actual update only after the availability of an actual update (versus a hash collision) is confirmed. The same arrangement can be employed in reverse.

21 Claims, 14 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 13

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. De

---

☐ 3. Document ID: US 6751794 B1

L2: Entry 3 of 24 File: USPT Jun 15, 2004

US-PAT-NO: 6751794
DOCUMENT-IDENTIFIER: US 6751794 B1

TITLE: Intelligent patch checker

DATE-ISSUED: June 15, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
McCaleb; Jed	San Francisco	CA		

Rive; Russel Palo Alto CA

US-CL-CURRENT: 717/168; 707/203, 709/201, 709/203, 709/220, 717/171, 717/172, 717/173, 717/176, 717/177, 717/178, 719/327

ABSTRACT:

A method to remotely update software for a plurality of client system is disclosed. A client system sends a request for an upgrade to a server system. The request includes a unique identification that is recognized by the server system as belonging to the client system. In response, the server system sends an instruction to the client system that directs the client system to collect application information about the software application installed on the client system. The client system sends the application information to the server system. The server system performs a comparison between the application information about the software application and the most-updated upgrade package for the software application. The server system sends the most-updated upgrade package for the software application to the client system.

59 Claims, 7 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

☐ 4. Document ID: US 6721942 B1

L2: Entry 4 of 24 File: USPT Apr 13, 2004

US-PAT-NO: 6721942
DOCUMENT-IDENTIFIER: US 6721942 B1
**** See image for Certificate of Correction ****

TITLE: Component object model interface to C++ type conversion

DATE-ISSUED: April 13, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Sievert; James A.	Lino Lakes	MN		

US-CL-CURRENT: 717/137; 717/116, 717/124, 717/164, 717/165, 717/167, 719/329

ABSTRACT:

Methods for converting from a COM interface pointer to an underlying C++ object are described in various embodiments. In the various embodiments, classes are constructed in support of the underlying C++ object. The classes are used to enforce rules safely convert a COM interface pointer. One rule is that a COM interface to be converted cannot be marshaled. A second rule is that an object requesting the C++ object must have legal access to the COM interface (for example, the same execution unit). A third rule is that the object type of the COM interface must be in the inheritance hierarchy of the C++ object.

21 Claims, 6 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 5. Document ID: US 6698018 B1

L2: Entry 5 of 24 File: USPT Feb 24, 2004

US-PAT-NO: 6698018
DOCUMENT-IDENTIFIER: US 6698018 B1

TITLE: System and method of multiple-stage installation of a suite of applications

DATE-ISSUED: February 24, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Zimniewicz; Jeff A.	Bellevue	WA		
Helgeson; Ryan	Kirkland	WA		
Marino; Phillip J.	Dublin	OH		
Johnson; Crista E.	Seattle	WA		

US-CL-CURRENT: 717/175; 717/169, 717/171, 717/176

ABSTRACT:

A multiple stage installation system for the installation and setup of a suite of applications segregates and organizes the preparation, installation, clean up, optimization, etc. into functional groupings that define the multiple stages of the installation process. These functional groupings include actions to be performed on behalf of and to any and all of the applications to be installed. While different stages may be defined, the system preferably includes a pre-install phase, an install phase, and a post-install phase during which different functional activities are performed. An optimization phase may also be included to allow optimization of applications that have already been installed. In a preferred embodiment, the applications to be installed implement a COM interface that contains a method for each stage supported. The core installation system determines the installation order for the applications, acquires the COM interface from the application, and for each install stage, calls the appropriate method on that interface. Each application's method for a given stage is called before any methods of any applications for the next stage.

14 Claims, 8 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 6. Document ID: US 6681391 B1

L2: Entry 6 of 24 File: USPT Jan 20, 2004

US-PAT-NO: 6681391
DOCUMENT-IDENTIFIER: US 6681391 B1

TITLE: Method and system for installing software on a computer system

DATE-ISSUED: January 20, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Marino; Phillip J.	Dublin	OH		
Winkler; David V.	Seattle	WA		
Johnson; Crista	Seattle	WA		
Nelson; William M.	Seattle	WA		

US-CL-CURRENT: 717/175; 711/133, 711/134, 711/154, 711/158, 711/165, 711/217, 717/174, 717/176, 718/102

ABSTRACT:

A method and system for installing software on a computer generates an installation order that ensures that a component required for the functioning of another component is already installed. Furthermore, it makes possible generating good installation orders to allow related components, e.g., in a software suite, to be installed close together, thus reducing disk swapping. The method and system take into account the existing configuration on a computer and allow removal of components along with dynamic reconfiguration of a computing system in response to a user's choice of an application program to launch. In accordance with the invention, preferably a developer includes information about the component's relationship with other components, e.g., a specific requirement for a preinstalled component or a requirement that a particular component not be present, thus requiring its removal. To remove the possibility of a single identifier referring to more than one component, the preferred embodiments of the invention use globally unique identifiers to label individual components.

40 Claims, 7 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☐ 7. Document ID: US 6668375 B1

L2: Entry 7 of 24 File: USPT Dec 23, 2003

US-PAT-NO: 6668375
DOCUMENT-IDENTIFIER: US 6668375 B1

TITLE: Method and system for providing build-to-order software applications

DATE-ISSUED: December 23, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Leovac; Dino	Allison Park	PA		

US-CL-CURRENT: 717/174; 380/286, 705/51, 705/64, 707/100, 707/203, 717/168, 717/173, 717/175, 717/176, 717/177, 717/178

ABSTRACT:

A system and corresponding method for unlocking options in already installed software. The software is provided with all of the options on an installation medium and when the software is first received only a basic version is installed. The options are requested by a user and the customer service system provides a key based on information about the user and about the options requested. The installation software constructs a key from the same information and compares the key received from customer service. If the keys match, the requested options are installed.

9 Claims, 3 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 3

☐ 8. Document ID: US 6651080 B1

L2: Entry 8 of 24 File: USPT Nov 18, 2003

US-PAT-NO: 6651080
DOCUMENT-IDENTIFIER: US 6651080 B1

TITLE: Techniques for implementing pluggable virtual machines

DATE-ISSUED: November 18, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Liang; Sheng	Mountain View	CA		
Lindholm; Timothy G.	Palo Alto	CA		

US-CL-CURRENT: 718/1; 717/148, 717/162, 717/163, 717/164, 717/165, 717/166, 719/331, 719/332

ABSTRACT:

Techniques for developing and exchanging virtual machine implementations and/or support library implementations are described. In one embodiment, the virtual machine design specifies a set of functions for executing all or substantially all support library operations that are dependent on the implementation of the virtual machine. When a developer desires to substitute one virtual machine implementation for another, the developer is able to basically "plug-in" the second virtual

machine implementation with minimal impact on the support libraries since both virtual machine implementations provide implementations for the set of specified functions that are dependent on the implementation of the respective virtual machine. Conversely, different support libraries may be utilized in conjunction with a particular virtual machine implementation.

21 Claims, 8 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

☐ 9. Document ID: US 6574729 B1

L2: Entry 9 of 24 File: USPT Jun 3, 2003

US-PAT-NO: 6574729
DOCUMENT-IDENTIFIER: US 6574729 B1

TITLE: System for remotely identifying and providing information of unknown software on remote network node by comparing the unknown software with software audit file maintained on server

DATE-ISSUED: June 3, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Fink; Paul Ernest	Mendham	NJ		
Henness; Marc A.	Saylorsburg	PA		
Szablowski; Walt	Newtown	PA		

US-CL-CURRENT: 713/1; 707/1, 707/10, 709/217, 717/171

ABSTRACT:

A method and apparatus are disclosed for remotely identifying software and software versions using a maintained software audit file. The disclosed system management tool (SMT) identifies software installed on each network node by comparing the name and size of installed files to a software audit file. The system management tool (SMT) performs an inventory scan of the software on each network node and obtains a list of each file and the corresponding file size. The software audit file provides identifying information, such as the file name and corresponding size, for each known file. Known files can be quickly identified using a match criteria based, for example, on the file name and size. The software audit file is maintained by investigating any unknown files with a sample of the user population having the unknown file. In one implementation, a targeted query is automatically transmitted to a sample of the user population having the unknown file requesting header information for the unknown file. In this manner, previously unknown files, once identified, can be added to the software audit file. A technique is disclosed for quickly identifying a network node, in order to retrieve a list of instructions to be executed by the network node.

18 Claims, 11 Drawing figures
Exemplary Claim Number: 1

Number of Drawing Sheets: 8

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 10. Document ID: US 6560776 B1

L2: Entry 10 of 24 File: USPT May 6, 2003

US-PAT-NO: 6560776
DOCUMENT-IDENTIFIER: US 6560776 B1

TITLE: Software installation verification tool

DATE-ISSUED: May 6, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Breggin; David G.	Littleton	CO		
Drapal; Myron Eugene	Lafayette	CO		
Prenger; Deborah K.	Boulder	CO		

US-CL-CURRENT: 717/176; 707/203

ABSTRACT:

The method and system of the present invention automatically generates an installation file or database containing information describing or characterizing the installation. A verifying tool can compare the installation information to installed information relating to or describing the files actually installed by the install program.

47 Claims, 11 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 11

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 11. Document ID: US 6546553 B1

L2: Entry 11 of 24 File: USPT Apr 8, 2003

US-PAT-NO: 6546553
DOCUMENT-IDENTIFIER: US 6546553 B1
** See image for Certificate of Correction **

TITLE: Service installation on a base function and provision of a pass function with a service-free base function semantic

DATE-ISSUED: April 8, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hunt; Galen C.	Bellevue	WA		

US-CL-CURRENT: 717/174; 712/233, 712/234, 712/244, 714/38, 717/163, 717/175, 717/176, 717/177

ABSTRACT:

A base function provides a base function semantic. During service installation, an unconditional branch instruction to a service function replaces one or more instructions at the beginning of a base function. The service function provides a service semantic such as instrumentation, redirection, replacement, or extension. After service installation, a pass function includes the replaced base function instructions and an unconditional branch instruction to the logically subsequent base function instruction. Thus, the pass function provides a service-free base function semantic. The service function calls the pass function an arbitrary number of times before and/or after executing any other service function instructions. The pass function is allocated statically or dynamically. A statically allocated pass function is callable before and/or after service installation to guarantee a service-free base function semantic. A service removal function restores a base function and conforms a pass function to the restored base function. A pass function is callable before and/or after service removal. A library of service installation functions includes functions for installing and removing a service on a base function. A library of binary editing functions includes functions for attaching service installation functions and associated data payloads to a binary file.

57 Claims, 9 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 9

☐ 12. Document ID: US 6496865 B1

L2: Entry 12 of 24 File: USPT Dec 17, 2002

US-PAT-NO: 6496865
DOCUMENT-IDENTIFIER: US 6496865 B1

TITLE: System and method for providing interpreter applications access to server resources in a distributed network

DATE-ISSUED: December 17, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Sumsion; John G.	Spanish Fork	UT		
Echols; Grant G.	Payson	UT		

US-CL-CURRENT: 709/229; 709/203, 709/225, 717/139, 717/163, 717/165, 717/166, 719/315

ABSTRACT:

A resource access system and method for providing interpreters with the ability to provide interpreter applications access to any desired server resource regardless of type of server and current capability of the client node to access such server resources. The invention is an application-level extension of the interpreter, enabling the interpreter to provide server resource access independently of the type of operating system implemented in the client node. In addition, the invention utilizes existing techniques to communicate with the server, such as a distributed object system or an existing client redirector, enabling the invention to provide such access with minimal modifications to the client or server nodes, accommodating the client node's current capability to access the server resources. Also, the resource access system provides access without having to use a foreign application interface.

15 Claims, 9 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 9

☐ 13. Document ID: US 6381742 B2

L2: Entry 13 of 24 File: USPT Apr 30, 2002

US-PAT-NO: 6381742
DOCUMENT-IDENTIFIER: US 6381742 B2
**** See image for Certificate of Correction ****

TITLE: Software package management

DATE-ISSUED: April 30, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Forbes; Jonathan A.	Bellevue	WA		
Stone; Jeremy D.	Bellevue	WA		
Parthasarathy; Srivatsan	Issaquah	WA		
Toutonghi; Michael J.	Seattle	WA		
Sliger; Michael V.	Issaquah	WA		

US-CL-CURRENT: 717/176; 707/203

ABSTRACT:

A software package manager uses a distribution unit containing components for a software package and a manifest file that describes the distribution unit to manage the installation, execution, and uninstallation of software packages on a computer. Information in the manifest file pertaining to a software package is stored in a code store data structure upon installation of the package. The manifest file also contains information that permits the software package manager to resolve any software dependencies upon installation. The software package manager uses the code store data structure to locate the required components when the software is

executed and to remove the components appropriately when the software is uninstalled.

43 Claims, 9 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 8

☐ 14. Document ID: US 6311321 B1

L2: Entry 14 of 24 File: USPT Oct 30, 2001

US-PAT-NO: 6311321
DOCUMENT-IDENTIFIER: US 6311321 B1

TITLE: In-context launch wrapper (ICLW) module and method of automating integration of device management applications into existing enterprise management consoles

DATE-ISSUED: October 30, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Agnihotri; Manoj B.	Lake Oswego	OR		
Tung; Chiaohuey (Peter)	Portland	OR		

US-CL-CURRENT: 717/120; 709/220, 710/302, 717/176

ABSTRACT:

Embodiments of the present invention are directed to an In-Context Launch Wrapper (ICLW) module which provides a comprehensive generic interface for automating integration of device management applications (applets) into existing Enterprise management console(s) installed at a host system of a network for centralized remote device management of remote network devices on a network. The ICLW module comprises a set of extensible software components, including: an Install component which provides a file based interface to device management applications for installation specific to the existing enterprise management consoles; a MConsole Interface component which provides a comprehensive generic interface to existing enterprise management consoles for enabling integration of device management applications into the existing enterprise management consoles; a Discovery component which provides a discovery interface to the existing enterprise management consoles for identifying remote network devices on a network; and a Trap Extension component which provides an extension interface to the existing enterprise management consoles for handling Simple Network Management Protocol (SNMP) traps on a network.

23 Claims, 6 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

☐ 15. Document ID: US 6279153 B1

L2: Entry 15 of 24 File: USPT Aug 21, 2001

US-PAT-NO: 6279153
DOCUMENT-IDENTIFIER: US 6279153 B1
**** See image for Certificate of Correction ****

TITLE: Multi-user flash ROM update

DATE-ISSUED: August 21, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bi; Depeng	Mt. Prospect	IL		
Wilson; James Y.	Crystal Lake	IL		

US-CL-CURRENT: 717/171

ABSTRACT:

A system in which a plurality of wireless interface devices, each containing one or more flash memory devices, are interfaced to a server which may be connected in either a wireless or wired LAN by way of a radio link. The system in accordance with the present invention, enables the flash or other type of memory devices in the plurality of wireless interface devices interfaced to the server to be updated over a radio link.

17 Claims, 222 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 126

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. D

☐ 16. Document ID: US 6192518 B1

L2: Entry 16 of 24 File: USPT Feb 20, 2001

US-PAT-NO: 6192518
DOCUMENT-IDENTIFIER: US 6192518 B1

TITLE: Method for distributing software over network links via electronic mail

DATE-ISSUED: February 20, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Neal; Douglas Edward	Howell	MI		

US-CL-CURRENT: 717/175; 709/221, 717/176, 717/178

ABSTRACT:

The present invention discloses a method, apparatus, and article of manufacture for distributing a software application residing on a network, from a source computer coupled to the network, to a remote computer via electronic mail.

At the source computer, the present invention identifies the software installation components that reside on the source computer. These software installation components are used to install the software application. The identified software installation components are compared to the remote computer's software installation components. This comparison determines which software installation components are missing from the remote computer. To provide the files required for installation of the software application, the present invention transfers the missing software installation components from the source computer to the remote computer.

9 Claims, 12 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 9

☐ 17. Document ID: US 6178548 B1

L2: Entry 17 of 24 File: USPT Jan 23, 2001

US-PAT-NO: 6178548
DOCUMENT-IDENTIFIER: US 6178548 B1

TITLE: Binary class library with debugging support

DATE-ISSUED: January 23, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hickman; Kevin Paul	Coquitlam			CA
McCrady; Donald James	North York			CA
Sarantakos; William	Willowdale			CA
Stoodley; Kevin Alexander	Richmond Hill			CA
Thomson; Brian Ward	North York			CA

US-CL-CURRENT: 717/124; 717/163, 717/165

ABSTRACT:

A binary class library is adapted to provide full debugging type information particularly for use during program compilation in a minimal debug-generation mode. The library includes, a compile unit that #includes all the include files describing the classes that a given class library implements and exports. The compile unit also includes code defining a symbol with an external linkage. The resulting object module is put into the binary class library or a separate debug library. An inclusion direction and an external linkage symbol to the debug library are added to all include files for the class library that are #included by any program using its classes. These additions cause the compiler to direct the linker

to add the debug library to the list of libraries from which it tries to resolve symbolic references, and to add a reference to the external linkage symbol that the linker will have to resolve by adding the debug library into the link.

17 Claims, 3 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 3

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. D

☐ 18. Document ID: US 5960204 A

L2: Entry 18 of 24 File: USPT Sep 28, 1999

US-PAT-NO: 5960204
DOCUMENT-IDENTIFIER: US 5960204 A
**** See image for Certificate of Correction ****

TITLE: System and method for installing applications on a computer on an as needed basis

DATE-ISSUED: September 28, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Yinger; Glenn S.	Parker	CO		
McVaney; Charles E.	Englewood	CO		
Foos; James L.	Highlands Ranch	CO		

US-CL-CURRENT: 717/176; 709/221

ABSTRACT:

A data processing system for computer application installation on a client/server network on an as needed basis is disclosed. The server computer includes an installation unit, a system application repository, and a user information file. The client computer includes a menu driver, an application check unit, a local application repository, and a local version repository. A user on the client computer selects an application to execute through the menu driver. The application check unit searches the local application repository and the local version repository for the application and, if found, determines whether it is the most current version unless the most current version is already installed on the client computer. After installation is completed on the client computer, the application is automatically executed. Further, the present invention includes a method for installing a computer applications and/or update on an as needed basis.

12 Claims, 13 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 12

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. D

☐ 19. Document ID: US 5898875 A

L2: Entry 19 of 24 File: USPT Apr 27, 1999

US-PAT-NO: 5898875
DOCUMENT-IDENTIFIER: US 5898875 A

TITLE: Method and computer system for loading objects

DATE-ISSUED: April 27, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Nakamura; Hiroaki	Yamato			JP
Onodera; Tamiya	Tokyo-to			JP
Takauchi; Mikio	Zama			JP

US-CL-CURRENT: 717/165; 717/116

ABSTRACT:

An objects loading method comprising the steps of (1) determining whether an object which is going to be loaded is a first object which is accessed only by another object, (2) if the object which is going to be loaded is determined to be the first object, removing the first object and also at least one second object which is accessed by the first object from the objects to be loaded, and (3) updating a list for managing the loaded objects.

6 Claims, 7 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. D

☐ 20. Document ID: US 5889992 A

L2: Entry 20 of 24 File: USPT Mar 30, 1999

US-PAT-NO: 5889992
DOCUMENT-IDENTIFIER: US 5889992 A

TITLE: Method for mapping types stored in a model in an object-oriented repository to language constructs for A C binding for the repository

DATE-ISSUED: March 30, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Koerber; Paul Donald	Fountain Valley	CA		

US-CL-CURRENT: 717/108; 717/165

ABSTRACT:

The method of the present invention is useful in a computer system having a user interface, a CPU, a memory, at least one disk drive, and an object-oriented repository, a program operating in the computer system for accessing the object-oriented repository. The program executes a method for mapping types in a model stored in the repository to language constructs for a C binding to the repository. The method first processes each type in the model, then the program processes each data type in the model. Function declarations and C to C++ wrapper functions are generated for each type and data type.

12 Claims, 12 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 12

☐ 21. Document ID: US 5867713 A

L2: Entry 21 of 24 File: USPT Feb 2, 1999

US-PAT-NO: 5867713
DOCUMENT-IDENTIFIER: US 5867713 A

TITLE: Committing an install plan object for the network installation of application programs

DATE-ISSUED: February 2, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Shrader; Theodore Jack London	Cedar Park	TX		
Bunce; John Lawrence	Austin	TX		
Jensen; Barbara Jean	Austin	TX		

US-CL-CURRENT: 717/176; 709/220, 717/175

ABSTRACT:

Committing an installation plan object for installing applications in a network. The installation plan object includes an application-in-plan object which represents an application program and a group-in-plan object which represents a group of workstations on which the application program is to be installed. As part of the commit process, the installation plan object is prevalidated by examining its child objects and adding additional child objects to the installation plan object if required, validated by examining data in the installation plan object and its child objects for errors in the data and transformed into data structures usable for a network installation engine which installs applications across a network. The installation plan further includes a response file object if the application's installation requires a response file and a customization file object which contains data to customize the response file object data for particular workstations.

25 Claims, 38 Drawing figures

Exemplary Claim Number: 1
Number of Drawing Sheets: 27

☐ 22. Document ID: US 5740469 A

L2: Entry 22 of 24 File: USPT Apr 14, 1998

US-PAT-NO: 5740469
DOCUMENT-IDENTIFIER: US 5740469 A

TITLE: Apparatus for dynamically reading/writing multiple object file formats through use of object code readers/writers interfacing with generalized object file format interface and applications programmers' interface

DATE-ISSUED: April 14, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Yin; Weiping	Austin	TX		
Hancock; Paul	Austin	TX		
Weiner; Alan	Austin	TX		

US-CL-CURRENT: 710/65; 345/520, 717/100, 717/136, 717/165

ABSTRACT:

An apparatus for allowing a single software Tool (136) to read and write multiple Object File Formats utilize dynamically configurable and loadable Object File Readers (131) and Writers (139). A separate Reader (131) and Writer (139) can be implemented for each different Object File Format and variations thereof. Tools (136) communicate with the Readers (131) and Writers (139) using a Generalized Object File Program Interface (124). This Interface (124) utilizes Data Structures implementing a Generalized Object File Internal Representation (122) and an Applications Programmers Interface (120).

8 Claims, 5 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 5

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☐ 23. Document ID: US 5682536 A

L2: Entry 23 of 24 File: USPT Oct 28, 1997

US-PAT-NO: 5682536
DOCUMENT-IDENTIFIER: US 5682536 A

TITLE: Method and system for referring to and binding to objects using identifier objects

DATE-ISSUED: October 28, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Atkinson; Robert G.	Woodinville	WA		
Williams; Antony S.	Mercer Island	WA		
Jung; Edward K.	Seattle	WA		

US-CL-CURRENT: 717/165; 719/315

ABSTRACT:

A method and system for referring to and binding to objects using a moniker object is provided. In a preferred embodiment, a moniker object contains information to identify linked source data and provides methods through which a program can bind to the linked source data. A binding method is provided that returns an instance of an interface through which the linked source data can be accessed. The moniker object can identify source data that is stored persistently or nonpersistently. In addition, moniker objects can be composed to form a composite moniker object. A composite moniker object is used to identify linked source data that is nested in other data. In a preferred embodiment, the moniker object provides other methods including a reducing method that returns a more efficient representation of the moniker object; equality and hash methods for comparing moniker objects; and inverse, common prefix, and relative-path-to methods for comparing and locating moniker objects from other moniker objects. Several implementations of a moniker object are provided including a file moniker, an item moniker, a genetic composite moniker, a pointer moniker, and an anti moniker. Each implementation is a moniker class and has a class identifier that identifies code to manage the moniker class.

6 Claims, 67 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 53

☐ 24. Document ID: US 5421016 A

L2: Entry 24 of 24 File: USPT May 30, 1995

US-PAT-NO: 5421016
DOCUMENT-IDENTIFIER: US 5421016 A

TITLE: System and method for dynamically invoking object methods from an application designed for static method invocation

DATE-ISSUED: May 30, 1995

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Conner; Michael H.	Austin	TX		

Coskun; Nurcan	Austin	TX
Martin; Andrew R.	Austin	TX
Raper; Larry K.	Austin	TX

US-CL-CURRENT: 717/146; 717/116, 717/165, 719/315, 719/317, 719/330

ABSTRACT:

A method, system and program for allowing an application designed to use static method calls to manipulate objects whose methods are only available through dynamic calls without modifying the binary image of the application. A SOM compiler generates class definitions and generates a redispatch stub for each method defined in a class. A redispatch stub is a short sequence of instructions with an identical calling sequence as its corresponding method. This gives the class' dispatch enough information to determine the correct method procedure in a dynamic manner. The dispatch function employs the redispatch stub to call the appropriate method procedure and return any return value to the calling application via the redispatch stub. Redispatch stubs allows a class with a definition that can vary at runtime to be used by an application that was designed for statically defined classes.

11 Claims, 15 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 8